

LA-UR- 09-06438

Approved for public release;  
distribution is unlimited.

Title: Phenix – A Comprehensive Python-based System for  
Macromolecular Structure Solution

Author(s): Paul D. Adams, LBNL, UC Berkeley; Pavel V. Afonine, UC Berkeley;  
Gábor Bunkóczi, Univ of Cambridge; Vincent B. Chen, Duke Univ;  
Ian Davis, Duke Univ; Nathaniel Echols, LBNL; Jeffrey J. Headd,  
Duke Univ; Li-Wei Hung, LANL; Ralf W. Grosse Kunstleve, LBNL;  
Airlie J. McCoy, Univ of Cambridge; Nigel W. Moriarty, LBNL;  
Robert Oeffner, Univ of Cambridge; Randy J. Read, Univ of  
Cambridge; David C. Richardson, Duke Univ; Jane S. Richardson,  
Duke Univ; Thomas C. Terwilliger, LANL; and Peter H. Zwart, LBNL

Intended for: Acta Crystallographica D  
International Tables of Crystallography



Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

# Phenix – A Comprehensive Python-based System for Macromolecular Structure Solution

Paul D. Adams,<sup>a,b</sup> Pavel V. Afonine,<sup>a</sup> Gábor Bunkóczi,<sup>c</sup> Vincent B. Chen,<sup>d</sup> Ian Davis,<sup>d,‡</sup> Nathaniel Echols,<sup>a</sup> Jeffrey J. Headd,<sup>d</sup> Li-Wei Hung,<sup>e</sup> Ralf W. Grosse-Kunstleve,<sup>a</sup> Airlie J. McCoy,<sup>c</sup> Nigel W. Moriarty,<sup>a</sup> Robert Oeffner,<sup>c</sup> Randy J. Read,<sup>c</sup> David C. Richardson,<sup>d</sup> Jane S. Richardson,<sup>d</sup> Thomas C. Terwilliger,<sup>e</sup> and Peter H. Zwart<sup>a</sup>

<sup>a</sup> Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA

<sup>b</sup> Department of Bioengineering, UC Berkeley, CA, 94720, USA

<sup>c</sup> Department of Haematology, University of Cambridge, Cambridge Institute for Medical Research, Wellcome Trust/MRC Building, Cambridge CB2 0XY, UK

<sup>d</sup> Department of Biochemistry, Duke University Medical Center, Durham, NC 27710, USA

<sup>e</sup> Los Alamos National Laboratory, Los Alamos, NM 87545, USA

<sup>‡</sup> Current Address: Department of Biochemistry, University of Washington, Seattle, WA 98195, USA

**Synopsis** The Phenix software for macromolecular structure determination is described.

**Abstract** Macromolecular X-ray crystallography is routinely applied to understand biological processes at a molecular level. However, significant time and effort are still required to solve and complete many of these structures because of the need for manual interpretation of complex numerical data using many software packages, and the repeated use of interactive three-dimensional graphics. Phenix has been developed to provide a comprehensive system for crystallographic structure solution with an emphasis on automation of all procedures. This has relied on the development of algorithms that minimize or eliminate subjective input, the development of algorithms that automate procedures that are traditionally performed by hand, and finally the development of a framework that allows a tight integration between the algorithms.

**Keywords:** Phenix, Python, C++, Crystallography, Algorithms

## 1. FOUNDATIONS

### 1.1. Phenix Architecture

The Phenix architecture is from the ground up designed as a hybrid system of tightly integrated interpreted (“scripted”) and compiled software modules. A mix of scripted and compiled components is found invariably in all major, successful crystallographic packages, but often the scripting is added as a second thought in an ad hoc fashion, using tools that

predate the object-oriented era. While such ad hoc systems are quickly established, they tend to become a severe maintenance burden as they grow. In addition users are often forced into many time-consuming routine tasks such as manually converting file formats. In Phenix, the scripting layer is the heart of the system. With only a few exceptions, all major functionality is implemented as modules that are exclusively accessed via the scripting interfaces. The object-oriented Python scripting language (Lutz & Ascher, 1999) is used for this purpose. In about two decades, a large developer/user community has produced millions of lines of highly uniform, interoperable, mature, and openly available sources covering all aspects of programming ranging from simple file handling to highly sophisticated network communication and fully featured cross-platform graphical interfaces. Embedding crystallographic methods into this environment enables an unprecedented degree of automation, stability, and portability. By design, the object-oriented programming model fosters shared collaborative development by multiple groups. It is routine practice to hierarchically recombine modules written by different groups into ever more complex procedures that appear uniform from the outside. A more detailed overview of key software technology leading to all these advances, presented in the context of crystallography, can be found in Grosse-Kunstleve et al. (2002).

In addition to the advantages outlined in the previous paragraph, the scripting language is generally most efficient for the rapid development of new algorithms. However, runtime performance considerations often dictate that numerically intensive calculations are eventually implemented in a compiled language. The first choice of a compiled language is of course to reuse the same language environment used for the scripting language itself, which is a C/C++ environment. Not only is this the mainstream software environment on all major platforms used today, but with probably hundreds of millions of lines of C/C++ sources in existence it is an environment that is virtually guaranteed to thrive long term. An in-depth discussion of the combined use of Python and C++ can be found in Grosse-Kunstleve et al. (2002) and Adams & Grosse-Kunstleve (2003). This model is used throughout the Phenix system.

## **1.2. Graphical User Interface**

A new graphical user interface (GUI) for Phenix was introduced in version 1.4. It uses the open-source wxPython toolkit, which provides a “native” look on each operating system. Development has focused on providing interfaces around the existing command-line programs with minimal modification, using the same underlying configuration system (libtbx.phil) used by most Phenix programs as a template to automatically generate controls. Because these programs are implemented primarily as Python modules, complex data including models, reflections, and other viewable data may be exchanged with the GUI

without resorting to parsing log files. The current Phenix release (1.5) includes GUIs for *phenix.refine*, *phenix.xtriage*, the AutoSol, AutoBuild, and LigandFit wizards, the restraints editor REEL, all of the validation tools, and several utilities for creating and manipulating maps and reflection files. More recent nightly builds of Phenix contain a new GUI for the AutoMR wizard, and future releases will include a new interface for Phaser (McCoy et al., 2007).

Intrinsically graphical data is visualized with embedded graphs (using the free matplotlib Python library) or a simple OpenGL viewer. This simplifies the most complex parameters, such as atom selections in *phenix.refine*, which can be visualized or picked interactively with the built-in viewer. The GUI also serves as a platform for additional automation and user customization. Similarly to the CCP4 interface (ccp4i; Potterton et al. 2003), Phenix manages data and task history for separate user-defined projects. Default parameters and input files can be specified for each project; for instance, generation of ligand restraints from the *phenix.refine* GUI gives the user the option of automatically loading these restraints in future runs.

The popularity of Python as a scientific programming language has led to its use in many other structural biology applications, especially molecular graphics software. The Phenix GUI includes extension modules for the modeling programs Coot (Emsley & Cowtan 2004) and PyMOL (DeLano 2002), both of which are controlled remotely from Phenix using the XML-RPC protocol. This allows the interfaces to integrate seamlessly; any model or map in Phenix can be automatically opened in Coot with a single click. In programs that iteratively rebuild or refine structures, such as AutoBuild and *phenix.refine*, the current model and maps will be continually updated in Coot and/or PyMOL as soon as they are available. For the validation utilities, clicking on any atom or residue flagged for poor statistics will recenter the graphics windows on that atom. Remote control of the Phenix GUI is also simple using the same protocol, and simple extensions to the Coot interface provide direct launching of *phenix.refine* with a model pre-loaded.

## 2. ANALYSIS OF EXPERIMENTAL DATA

Phenix has a range of tools for the analysis, validation, and manipulation of X-ray diffraction data. A comprehensive tool for analyzing X-ray diffraction data is Xtriage (Zwart et al., 2005), which carries out tests ranging from space group and twinning to detection of anomalous signal. These tests provide the user and the Wizards (Terwilliger et al, 2009) with a set of statistics that characterize a dataset. For the analysis of twinning, *phenix.xtriage* consolidates a number of statistics to provide a balanced verdict of possible symmetry and twin-related issues with the data. *Phenix.xtriage* provides the user with feedback on the overall characteristics of the data. Routine usage of *phenix.xtriage*, during or immediately

after data collection, has resulted in the timely discovery of twinning or other issues (Flynn et al., 2007; Kostecky et al., 2009). Detection of these idiosyncrasies in the data typically reduces the overall effort in a successful structure determination.

A likelihood-based estimation of the overall anisotropic scale factor is performed using the likelihood formalism described by Bourenkov & Popov, 2003. Database derived, standard Wilson plots for proteins and nucleic acids are used to detect anomalies in the mean intensity. These anomalies can be due to ice rings or other issues (Morris et al., 2004). Data strength and low resolution completeness is also analysed. The presence of anomalous signal is detected by analysis of the measurability, a quantity expressing the fraction of statistically significant Bijvoet differences in a data set (Zwart, 2005). The native Patterson function is used to detect the presence of pseudo translational symmetry. A database-derived empirical distribution of maximum peaks heights is used to assign significance to detected peaks in the Patterson function.

A comprehensive automated twinning analysis is performed. Twin laws are derived from first principles to facilitate identification of pseudo-merohedral cases. Amplitude and intensity ratios,  $\langle |E^2 - 1| \rangle$  values, the L-statistic (Padilla & Yeates, 2003), and NZ plots are derived from data cut to the resolution limit suggested by the data strength analysis. Removal of shells of data with relatively high noise content greatly improves the automated interpretation of these statistics. A Britton plot, H-test and a likelihood-derived approach, are used to estimate twin fractions when twin laws are present. If a model has been supplied, an R vs R (Lebedev et al., 2006) analysis is carried out. This type of analysis is of particular use when dealing with pseudo symmetry, space group problems and twinning (Zwart et al., 2008).

To test for inconsistent indexing between different data sets, a set of reindexing laws is derived from first principles given the unit cells and space groups of the sample and reference data sets. A correlation analysis suggests the most likely choice of reindexing the data. Analysis of the metric symmetry of the unit cell provides a number of likely point groups. A likelihood-inspired method is used to suggest the most likely point group of the data. Subsequent analysis of systematic absences in a likelihood framework ranks subsequent space group possibilities (details to be published).

### **3. SUBSTRUCTURE DETERMINATION, PHASING, AND MOLECULAR REPLACEMENT**

After ensuring that the diffraction data are sound and understood, the next critical necessity for solving a structure is the determination of phases, by one of several strategies.

#### **3.1. SUBSTRUCTURE DETERMINATION**

The substructure determination procedure implemented as *phenix.hyss* (Hybrid Substructure Search) combines the multi-trial dual-space recycling approaches pioneered by



Shake-and-Bake (Miller *et al.*, 1994) and later SHELXD (Schneider & Sheldrick, 2002) with the use of the fast-translation function (Navaza & Vernoslova, 1995, Grosse-Kunstleve & Brunger, 1999). The fast-translation function is the basis for a systematic search in the Patterson function (performed in reciprocal space), in contrast to the stochastic alternative of SHELXD (performed in direct space). *Phenix.hyss* is the only substructure determination program to fully integrate automatic comparison of the substructures found in multiple trials via an Euclidean Model Matching procedure (part of the *cctbx* open-source libraries). This allows *phenix.hyss* to detect if the same solution was found multiple times, and to terminate automatically if this is the case. Extensive tests with a variety of SAD datasets (Grosse-Kunstleve & Adams, 2003) have lead to a parameterization of the procedure that balances runtime considerations and the likelihood that repeated solutions present the correct substructure. In many cases the procedure finishes in seconds if the substructure is detectable from the input data.

### 3.2. PHASING

Phaser, available in Phenix as *phenix.phaser*, applies the principle of maximum likelihood to solving crystal structures by molecular replacement, single-wavelength anomalous diffraction (SAD), or the combination of both. The likelihood targets take proper account of the effects of different sources of error (and, in the case of SAD phasing, their correlations), and allow different sources of information to be combined. In solving a molecular replacement problem with a number of different components, the information gained from a partial solution increases the signal in searching for subsequent components. Because the likelihood scores for different models can be directly compared, decisions among models can readily be made as part of automation strategies (discussed below).

### 3.3. NON-CRYSTALLOGRAPHIC SYMMETRY (NCS)

Non-crystallographic symmetry is an important feature of many macromolecular crystals that can be used to greatly improve electron density maps. Phenix has tools for the identification of NCS and for using NCS and multiple crystal forms of a macromolecule in phase improvement.

*Phenix.find\_ncs* and *phenix.simple\_ncs\_from\_pdb* are tools for identification of non-crystallographic symmetry in a structure using information from a heavy-atom substructure or an atomic model. *Phenix.simple\_ncs\_from\_pdb* will identify NCS and generate transformations from the chains in a model in a PDB file. *Phenix.find\_ncs* will identify NCS from either a heavy-atom substructure (Terwilliger, 2002a) or the chains in a PDB file, then it will compare this NCS with the density in a map to verify that the NCS is actually present.

*Phenix.multi\_crystal\_average* is a method for combining information from several crystal forms of a structure. It is especially well suited to cases where each crystal form has its own NCS, adjusting phases for each crystal form so that all the NCS copies in all crystals are as similar as possible.

NCS restraints should normally be applied in density modification and model-building in all cases except where there is clear evidence that NCS is not present. In density modification within Phenix the presence of NCS is identified from the heavy-atom sites or from an atomic model if available. The local correlation of density in NCS-related locations is then used automatically to set variable restraints on NCS symmetry in the map. In refinement, NCS symmetry is applied through coordinate restraints, targeting the positions of each NCS copy relative to those of the other NCS-related chains. The default NCS restraints in Phenix are very tight, with targets of 0.05 Å rms. At resolutions lower than about 2.5 Å these tight restraints on NCS should usually be applied. At higher resolutions, it may be appropriate to use looser restraints or to remove them altogether. Additionally, if there are segments of the chains that clearly do not obey the NCS relationships they should be excluded from the NCS restraints. Normally this is done automatically but can be specified explicitly as well.

#### 4. MODEL-BUILDING, LIGAND FITTING, AND NUCLEIC ACIDS

Key steps in the analysis of a macromolecular crystal structure are building an initial core model, identification and fitting of ligands into the electron density map, and building an atomic model for loop regions that are less well-defined than the majority of the structure. Phenix has tools for rapid model-building of secondary structure and main-chain tracing (*phenix.find\_helices\_strands*) and for the fitting of flexible ligands (*phenix.ligandfit*), as well as for fitting a set of ligands to a map (*phenix.find\_all\_ligands*) and for the identification of ligands in a map (*phenix.ligand\_identification*). Phenix additionally has a tool for fitting of missing loops (*phenix.fit\_loops*). Validation tools are provided so that the models produced can be validated at each step along the way.

##### 4.1. MODEL-BUILDING

*Phenix.find\_helices\_strands* will rapidly build a secondary structure-only model into a map, or very rapidly trace the polypeptide backbone of a model into a map. To build secondary structure in a map, *phenix.find\_helices\_strands* identifies  $\alpha$ -helical regions and  $\beta$ -strand segments, models idealized helices and strands into the corresponding density, allowing for bending of the helices and strands, and assembles these into a composite model. To very rapidly trace the main-chain in a map, *phenix.find\_helices\_strands* finds points along ridgelines of high density where C $_{\alpha}$  atoms might be located, identifies pairs and then triples of these C $_{\alpha}$  atoms that have density between the atoms and plausible geometry, constructs all

possible connections of these  $C_{\alpha}$  atoms into 9-mers, then identifies all the longest possible chains that can be made by joining the 9-mers. This process can build a  $C_{\alpha}$  model at a rate of about 20 residues/second, yielding a backbone model that can readily be interpreted visually or automatically to evaluate the quality of the map it is based on.

*Phenix.fit\_loops* will fit missing loops in an atomic model. It uses RESOLVE model-building (Terwilliger, 2003) to extend the chain from either end where a loop is missing and to connect the chains into a loop with the expected number of residues.

## 4.2. LIGAND FITTING

*Phenix.ligandfit* is a tool for fitting a flexible ligand into an electron density map (Terwilliger, et al., 2006). The key approaches used are to break the ligand into its component rigid body parts, to find where each of these can be placed into density, tracing the remainder of the ligand based on the positions of these core rigid body parts, and recombining the best parts of multiple fits, while scoring based on the fit to the density.

*Phenix.find\_all\_ligands* is a tool for finding all the instances of each of several ligands in an electron density map. *Phenix.find\_all\_ligands* finds the largest contiguous region of unused density in a map and uses *phenix.ligandfit* to fit each supplied ligand into that density. It then chooses the ligand that has the highest real-space correlation to the density (Terwilliger et al., 2007a). Then it repeats this process until no ligands can be satisfactorily fit into any remaining density in the map.

*Phenix.ligand\_identification* is a tool for identifying what ligands are compatible with unknown electron density in a map (Terwilliger et al., 2007a). It can search using the 200 most common ligands from the PDB or from a user-supplied list of ligands. *Phenix.ligand\_identification* uses *phenix.ligandfit* to fit each ligand to the map and identifies the best-fitting ligand using real-space correlation and surface complementarity of the ligand and the atoms in the structure surrounding the ligand binding site.

## 4.3. RNA AND DNA

In common with most crystallographic tools, Phenix was originally developed with protein structures primarily in mind. Now that nucleic acids, and especially RNA, are increasingly important in large biological structures, the system is being modified in places where subtle differences in procedure are needed rather than just the relevant libraries.

Model building in *phenix.autobuild* now has a preliminary set of nucleic acid procedures that take advantage of the relatively well determined phosphate and base positions, as well as the preponderance of double helix, and that make use of the RNA backbone conformers recently defined by the RNA Ontology Consortium (Richardson et al., 2008). Nucleic acid structures benefit significantly from torsion-angle refinement, recently added to the options in



*phenix.refine*. A principal problem in RNA models is getting the ribose pucker correct, although it is known to consist almost entirely of either C3'-endo (commoner, and the one in A-form helix) or C2'-endo (Altona and Sundaralingam, 1972). MolProbity uses the perpendicular distance from the 3' phosphate to the line of the C1'-N1/9 glycosidic bond as a reliable diagnostic of ribose pucker (Davis et al., 2007; Chen et al., 2009). That same test has now been built into *phenix.refine*, to allow the use of pucker-specific target parameters for bond lengths, angles, and torsions (Gelbin et al., 1996) rather than the uneasy compromise values (Parkinson et al., 1996) used in most pucker-agnostic refinement. Currently, if an incorrect pucker is diagnosed it must usually be fixed by user rebuilding, for instance in Coot (Emsley and Cowtan, 2004) or in RNABC (Wang et al., 2008). A rebuilding functionality will probably be incorporated into Phenix soon, but in the meantime the refinement will now correctly maintain the geometry of a C2'-endo pucker once it has been built and identified using conformation specific residue names.

#### 4.4. MAPS, MODELS, AND AVOIDING BIAS

*Phenix.refine* (and the graphical tool *phenix.create\_maps*) can produce various types of maps, including anomalous difference map (see for example: Boisvert et al., 1996; Roach, 2003), maximum-likelihood weighted  $(p \cdot mF_{\text{obs}} - q \cdot DF_{\text{model}}) \exp(i\alpha_{\text{model}})$  and regular  $(p \cdot F_{\text{obs}} - q \cdot F_{\text{model}}) \exp(i\alpha_{\text{model}})$ , where  $p$  and  $q$  are any user-defined numbers, filled and kick maps. The coefficients  $m$  and  $D$  of likelihood-weighted maps (Read, 1986) are computed using test set reflections as described in Lunin & Skovoroda (1995) and Urzhumtsev et al., (1996).

Data incompleteness, especially systematic incompleteness, can cause map distortions (Lunin, 1988; Urzhumtsev et al., 1989; Lunin & Skovoroda, 1991; Tronrud, 1996b; Lunina et al., 2002). An approach to remedying this problem is to replace ('fill') missing observations with non-zero values. One can use  $DF_{\text{model}}$  (similarly to REFMAC; Murshudov et al., 1997) to replace missing  $F_{\text{obs}}$ , or  $\langle F_{\text{obs}} \rangle$  where the  $F_{\text{obs}}$  are averaged out in a resolution bin around the missing  $F_{\text{obs}}$  value. Based on a limited number of tests, both 'filling' schemes produce similar results reiterating the importance of phases. However, it is important to keep in mind that by replacing missing  $F_{\text{obs}}$  there is a risk of introducing bias, and obviously the more the data is incomplete the more the risk is. At present it is advisable to use both maps simultaneously: filled and not filled.

An average kick map (AK map; Gunčar et al., 2000; Turk, 2007; Pražnikar et al., 2009) is the result of the following procedure: a large ensemble of structures is created where the coordinates of each structure from the ensemble are all randomly shaken. Then for each structure a map is computed. Finally, all maps are averaged to generate one averaged kick map. An AK map is expected to have less bias, less noise, enhance existing signal and can potentially clarify some initially bad densities.

A computationally intensive but powerful method of creating a very low-bias map is to carry out iterative model-building and refinement while omitting one region of the map from all calculations of structure factors (Terwilliger et al., 2008). The *phenix.autobuild* iterative-build OMIT map procedure carries this out automatically for either a single OMIT region or for overlapping OMIT regions to create a composite iterative-build OMIT map.

## 5. MODEL, AND MODEL-TO-DATA, VALIDATION

The result of crystallographic structure determination is the atomic model. There are three principal components in assessing the model quality: quality of the model geometry and sterics, and quality of model-to-data fit in real space and in reciprocal space. All three provide overall measures, and the first two aspects also provide checks for local outliers. [Validation of the experimental data was described in section 2 above.] Phenix includes many individual tools for specific aspects of validation, plus several systems that combine those results into overall summaries. Validation is provided both for user evaluation of the progress and results of a structure solution, and also to help inform automated choices made by other parts of the system.

Most aspects of the MolProbity model-validation tools (Davis et al., 2007; Chen et al., 2009) have been adapted or rewritten for integrated use within Phenix, and are presented to the user by the new GUI [section 1.2]. Hydrogen atoms are added by *phenix.reduce*, with optimization of entire local H-bond networks, consideration of the first layer of crystallographic waters, and optional correction of sidechain amide or histidine 180° "flips" (Word et al., 1999b). All-atom contacts (Word et al., 1999a) are calculated by *phenix.probe*, which provide the atomic overlap information needed for validation of serious all-atom steric clashes and can also be visualized in Coot. For the Phenix GUI, the set of MolProbity-based tools provide both overall model statistics, such as clashscore and percent outliers, and detailed lists of the Ramachandran (Lovell et al., 2003), rotamer (Lovell et al., 2000), C $\beta$  deviation (Lovell et al., 2003), and clash outliers. Commandline tools are available for these validation methods: *phenix.rotalyze*, *phenix.ramalyze*, *phenix.cbetadev*, *phenix.clashscore*, *phenix.reduce*, and *phenix.probe*. Additionally, *phenix.validate\_model*, which analyzes the deviations of bond-lengths, bond-angles, planarity, etc. from ideal library values, compliments the MolProbity torsional and steric tools.

*Phenix.real\_space\_correlation* asserts the local model-to-data correspondence by providing a quantitative measure of how the atomic model fits the electron density map at the residue or atom level (depending on the resolution). Rapidly obtaining a snapshot of global figures-of-merit for a crystallographic model and associated experimental data is a frequent task performed at all stages of structure solution. This task can be complicated for several reasons: presence of novel ligands or non-standard residues in the PDB-format (Berman et al.,

2000) coordinate file, data collected from twinned crystals, various reflection data file formats, different representation of atomic displacement parameters in the presence of TLS, experimental data type (X-ray, neutron, or joint), files with multiple models, and various formatting issues. *Phenix.model\_vs\_data* is designed to automatically handle all these complications with minimal user input (a PDB file and a reflection data file) and provide a concise summary output. *Phenix.polygon* (Urzhumtseva et al, 2009) is a graphical tool that is designed to indicate the similarity of validation parameters, such as free-R-value, for a particular structure compared to those deposited in the PDB. This comparison is performed for all other structures solved at similar resolution limits. The result is presented graphically. *Phenix.validation* combines all of the tools described above in one GUI, providing a single place for assessing the results of structure determination.

## 5.1. MODEL AND STRUCTURE FACTOR MANIPULATION AND ANALYSIS

Phenix has a range of tools for displaying, analyzing, and manipulating structure factor and model information. *Phenix.mtz.dump* and *phenix.cif\_as\_mtz* display and convert structure factor data. *Phenix.print\_sequence*, *phenix.pdb\_atom\_selection*, and *phenix.pdbtools* display and manipulate coordinate files.

*Phenix.tls* is a tool for extraction and manipulation of TLS information. Using this tool, TLS matrices and selections can be extracted from REFMAC or Phenix formatted PDB file headers and the total or residual atomic B-factors can be computed and output. Future functionality will include the complete analysis of TLS matrices and their graphical visualization.

*Phenix.get\_cc\_mtz\_mtz* and *phenix.get\_cc\_mtz\_pdb* are tools for analyzing the agreement between maps based on a pair of MTZ files, or between maps calculated from an MTZ file and a PDB file. The key attributes of these tools are that they automatically search all allowed origin shifts that might relate the two maps and that they write out a modified version of one of the MTZ files or of the PDB file, shifted to match the other.

## 6. STRUCTURE REFINEMENT

*Phenix.refine* is the state-of-the-art crystallographic structure refinement engine of Phenix. The foundational refinement machinery is a combination of highly efficient programming tools and new or re-thought crystallographic algorithms. *Phenix.refine* possesses an extensive set of tools that cover a majority of refinement scenarios, at any data resolution from low to ultra-high. Various reflection data formats (for example, CNS, MTZ, and SHELX) are recognized automatically. The input experimental data is checked for outliers (Read, 1999; Zwart et al., 2005) and any reflections identified are excluded from the refinement calculations. Twinning can also be taken into account by providing a twin law operator, which

can be obtained using *phenix.xtriage*. Both X-ray and/or neutron diffraction data can be used, and an option for joint XN refinement is available (simultaneous refinement against X-ray and neutron data). Each refinement run begins with robust mask-based bulk-solvent correction and anisotropic scaling (Afonine et al, 2005). Tools such as efficient rigid body refinement (multiple-zones algorithm; Afonine et al, 2009), simulated annealing refinement (Brunger et al, 1987) in Cartesian or torsion angles space (Grosse-Kunstleve et al, 2009), automatic NCS detection and its use as restraints in refinement, are important at low resolution and at the initial stages of refinement. A broad range of atomic displacement parameterizations are available, including grouped isotropic, constrained anisotropic (TLS), individual atomic isotropic or anisotropic, allowing efficient modelling of atomic displacement parameters at any resolution. Occupancy refinement (grouped, individual, group constrained for alternative conformations or any mixture) can be performed for any user-defined atoms. Atoms in alternative conformations are recognized automatically based on altLoc identifiers in the input PDB file and their occupancies are refined by default. Ordered solvent (water) model updating is integrated into the refinement process.

The availability of ultra-high resolution data makes it possible to visualize the residual density due to bonding effects; *phenix.refine* employs a novel Inter Atomic Scatterers model (Afonine et al, 2007) to adequately account for these features. A flexible parameterization of hydrogen atoms allows their use at any resolution, from sub-atomic (where their parameters can be refined individually) to low resolution (where a riding model is used). Refinement can be performed using a variety of refinement target functions, including maximum-likelihood, maximum-likelihood with experimental phase information, and amplitude least-squares. The refinement of coordinates can be performed in real or reciprocal space (allowing for dual-space refinement). Novel ligands can be easily included in refinement by providing a corresponding CIF file as an input (the CIF file can be automatically created using *phenix.ready\_set*).

Manual fixing of amino acid side chain rotamers can be time consuming especially for large structures. Although the use of simulated annealing refinement increases the convergence radius, it can still fail to fit incorrectly modelled side chains into the correct density. *Phenix.refine* has an option for automatic selection of the best rotamer based on a rotamer library (Lovell et al., 2000) and optimal fit into the density (details to be published elsewhere). Further, coupling real-space refinement with the built-in rotamer library and available MolProbity tools allows for automated identification and robust correction of the common systematic errors involving backward-fit conformations for Leu, Thr, Val, Ile, and Arg sidechains, as developed and tested in the Autofix method (Headd et al., 2009).

*Phenix.refine* allows for multi-step complex refinement protocols, where most of the available refinement strategies can be combined with each other and applied to any selected part of the model. For example, a run of *phenix.refine* may perform rigid body refinement, simulated annealing, individual and grouped B-factors combined with TLS refinement, constrained occupancy refinement, and automatic water picking.

The output of *phenix.refine* includes various maps (maximum-likelihood weighted, kicked, incompleteness corrected, anomalous difference, and those with any user-defined coefficients), complete model and data statistics, and PDB file with a formatted REMARK 3 header ready for PDB deposition. The *phenix.refine* GUI is integrated with COOT and PyMol, allowing for seamless visual analysis of the refined model and associated maps.

*Phenix.refine* is tightly integrated with other Phenix components making structure solution, building and refinement a one step process (for example, in the Automr and Autobuild wizards). It is routinely tested by automatic re-refinement of all models in the PDB for which the experimental data is available.

### **6.1. Ligand coordinate and restraint geometry generation**

The electronic Ligand Builder and Optimisation Builder (eLBOW) (Moriarty, 2009) is a suite of tools designed for the reliable generation of Cartesian coordinates and geometry restraints for both novel and known ligands. In line with the rest of the Phenix package, the eLBOW modules are written in Python with the numerically intensive portions of the code written in C++. eLBOW is a flexible platform for converting a majority of the common chemical inputs to an optimised 3-D coordinates and geometry restraints for refinement. Ligand geometries can be minimised using the semi-empirical AM1 quantum chemical method (Stewart, 2004), a numerically efficient and chemically accurate technique for the class of molecules commonly bound with proteins.

In addition, a graphical user interface for editing geometry restraints and simple geometry manipulation of ligands has been developed. The Restraints Editor, Essentially Ligands (REEL) removes the tedium of manually editing a restraints file by providing a number of the commonly performed actions via pull-down menus and other interactive features. The effect of changes in the restraints can be immediately reflected in the molecule view to provide user feedback.

A tool that uses many of the features of eLBOW to quickly and easier prepare a protein model for refinement is known as *ReadySet!* The flexibility of the python interface is exemplified by the use of Reduce, eLBOW and several smaller portions of the *cctbx* toolkit to add hydrogen and/or deuterium atoms to the model and ligands, generate metal coordination files and geometry restraints for unknown ligands. The files required for covalently bound ligands are also generated.



## 7. INTEGRATED STRUCTURE DETERMINATION

### 7.1. WHY AUTOMATION?

Automation has dramatically changed macromolecular crystallography over the past decade, both by greatly speeding up the process of structure solution, model-building, and refinement and by bringing the tools for structure determination to a much wider group of scientists. As automation becomes increasingly comprehensive, it will allow users to test many more possibilities for structure determination, it will allow improved estimation of uncertainties in the final structures, and it will allow the determination of ever more complex and difficult structures.

The Phenix environment has been developed with automation as a key and defining feature. Each tool within Phenix can seamlessly and nearly effortlessly be incorporated as part of any other tool or process in Phenix. This means that very complex tasks can be built up from well-tested and characterized tools, and that tools and higher-level methods can be re-used in many different contexts. With a full automatic regression testing system as an integral part of the Phenix environment, all these tasks and high-level methods are tested nightly to ensure the integrity of the entire Phenix system.

### 7.2. AUTOMATED STRUCTURE SOLUTION

Phenix has fully integrated structure solution capability for both experimental phasing (MAD, SAD, MIR, and combinations of these), carried out by *phenix.autosol*, and for molecular replacement, performed by *phenix.automr*. Each of these automated procedures feeds directly into the iterative model-building, density modification, and refinement of *phenix.autobuild*.

*Phenix.autosol* is designed to allow complete automation of experimental phasing, while allowing a high degree of flexibility for advanced users. Beginning with structure factor amplitudes and the sequence of the macromolecule, *phenix.autosol* uses *phenix.solve* (Terwilliger & Berendzen, 1999) to scale all datasets, *phenix.xtriage* (Zwart, Grosse-Kunstleve & Adams, 2005) to analyze the data for twinning and to correct any anisotropy in the data, and *phenix.hyss* (Grosse-Kunstleve & Adams, 2003) to find potential heavy-atom or anomalously-scattering atoms. *Phenix.autosol* carries out experimental phasing with *phenix.phaser* (McCoy et al., 2004, McCoy et al., 2007) or *phenix.solve* (Terwilliger & Berendzen, 1999), density modification with *phenix.resolve* (Terwilliger, 2000), and preliminary model-building using the methods in *phenix.autobuild* (Terwilliger et al, 2007b)

A key step in automated structure solution is the identification of which of several possible space-group and heavy-atom or anomalously-scattering atom substructures is correct. *Phenix.autosol* uses a Bayesian scoring algorithm based on analysis of the experimental

electron density maps to identify which substructures lead to the best maps (Terwilliger et al., 2009). The main features of the maps that are used in this evaluation are the skewness of the electron density (non-Gaussian histogram of density with more density in the positive tail than the negative one), and the correlation of local rms density (large, contiguous regions of high variation where the molecule is located, separate large, contiguous regions of low variation where the solvent is located).

*Phenix.autosol* is highly flexible, allowing any combination of experimental data, such as MAD+SIRAS or several SAD datasets. Although it is fully automated, the user can control nearly all aspects of the operation of the procedure, including the scoring criteria and decisions about how certain *phenix.autosol* should be that the correct solution is contained in the current lists of solutions.

*Phenix.autosol* can carry out phasing using a combination of experimental SAD data and molecular replacement information. If a molecular replacement model is available, *phenix.autosol* will use *phenix.phaser* (McCoy et al., 2004, McCoy et al., 2007) to complete the anomalous substructure iteratively by constructing log-likelihood gradient maps for the anomalous scatterers based on the model of the non-anomalous structure and any anomalous scatterers that have already been found. Then the anomalous substructure is used along with the model to calculate phases with *phenix.phaser*.

*Phenix.automr* carries out automated likelihood-based molecular replacement using *phenix.phaser* (Read, 2001, McCoy et al., 2005, McCoy, 2007, McCoy et al., 2007). The procedure is highly automated, allowing for several copies of each of several components to be placed in a single run, which can also test different possible choices of space group. If there are alternative choices of model for a component, the molecular replacement calculation can try each of them in turn, or combine them as a statistically-weighted ensemble. Although the evaluation of the likelihood targets is slow (Read, 2001), the use of fast approximations for the rotation search (Storoni et al., 2004) and the translation search (McCoy et al, 2005) gives run times competitive with traditional Patterson-based methods. Likelihood has been demonstrated to be more sensitive to the correct solution, particularly in difficult cases (Read, 2001). When there are several copies or several components to place, the ability of the likelihood functions to take advantage of preliminary partial solutions can give a crucial increase in the signal.

### 7.3. ITERATIVE MODEL BUILDING, DENSITY MODIFICATION, AND REFINEMENT

*Phenix.autobuild* is a highly integrated and automated procedure for model-building and model improvement through iterative model-building, density modification and refinement. *phenix.autobuild* uses *phenix.resolve* (Terwilliger, 2003a, Terwilliger, 2003b) to carry out model-building, model extension, model assembly, loop fitting, and building outside existing

models. It further uses *phenix.resolve* to improve electron density maps with statistical density modification (2000), including information from the newly-built models as well as that obtained from experiment (e.g., *phenix.autosol*), from NCS (Terwilliger, 2002b), and from other expected features of electron-density maps such as a flat solvent (Wang, 1985), presence of secondary structural features (Terwilliger, 2001), and presence of local patterns of density characteristic of macromolecules (Terwilliger, 2003c). To reduce model bias in the procedure prime-and-switch phasing can be used as well (Terwilliger, 2004). *Phenix.autobuild* uses *phenix.refine* (Afonine et al., 2005) throughout this process to improve the quality of the models that are built.

*Phenix.autobuild* provides two complementary approaches to model-building. For cases in which no model or only a preliminary model has been built, *phenix.autobuild* will construct a new model, considering the main chain of any supplied models as potential coordinates. In cases where a nearly-final model is available, *phenix.autobuild* can apply a *rebuild\_in\_place* approach in which the polypeptide chain is rebuilt, a few residues at a time, without changing the register or the overall features of the model.

The *rebuild\_in\_place* approach in *phenix.autobuild* provides a powerful method for assessment of uncertainties in an atomic model by repetitive rebuilding of the model, using different random seeds for each iteration (Terwilliger et al., 2007c). The variability in coordinates of each atom in the ensemble that is created is a lower bound on the uncertainty of the position of that atom.

## 8. Conclusions

Advances in computational methods and algorithms have made it possible to automate the solution of many structures with Phenix. However, many challenges still exist. In particular the development of automated methods that can be applied at low resolution (worse than 3.0Å) remains a priority. In this resolution range there are typically too few experimental data to uniquely define the macromolecular structure for automated *ab initio* model building. Thus, methods are required that rely on prior knowledge from existing macromolecular structures to permit productive automated data interpretation. These methods will need to be developed and applied for all stages of structure solution and tightly integrated to maximize the information extracted from the experimental data.

**Acknowledgements** The authors would like to thank the NIH (grant GM063210) and the Phenix Industrial Consortium for support of the Phenix project. This work was supported in part by the US Department of Energy under Contract No. DE-AC02-05CH11231. RJR is supported by a Principal Research Fellowship from the Wellcome Trust (UK). We are grateful to Luc Bourhis and Marat Mustyakimov, for contributions to the computational

crystallography toolbox and Phenix; Paul Emsley and Bernhard Lohkamp for their guidance interfacing Phenix with Coot; the CCP4 developers for making the CCP4 library available for inclusion in Phenix; Kevin Cowtan for XXX; Garib Murshudov and Alexei Vagin for making the monomer library available; and our colleagues in the crystallographic community for numerous scientific discussions.

## References

- Abrahams D., Grosse-Kunstleve R.W. (2003). Building Hybrid Systems with Boost.Python. *C/C++ Users Journal* 21, 29-36.
- Adams, P. D., Grosse-Kunstleve, R. W., Hung, L.-W., Ioerger, T. R., McCoy, A. J., Moriarty, N. W., Read, R. J., Sacchettini, J. C., Sauter, N. K. & Terwilliger, T. C. (2002). *Acta Cryst. D* 58, 1948-1954.
- Adams, P. D., Afonine, P., Grosse-Kunstleve, R. W., Read, R. J., Richardson, J. S., Richardson, D. C. & Terwilliger, T. C. (2009). *Curr. Opin. Struct. Biol.* published on-line, doi:10.1016/j.sbi.2009.07.014
- Afonine, P.V., Grosse-Kunstleve, R.W. and Adams, P.D. (2005). The Phenix refinement framework. *CCP4 Newsletter* July, Contribution 8.
- Altona, C., & Sundaralingam, M. (1972) *J. Am. Chem. Soc.* 94: 8205.
- Arendall, W. B. III, Tempel, W., Richardson, J. S., Zhou, W., Wang, S., Davis, I. W., Liu, Z.-J., Rose, J. P., Carson, W. M., Luo, M., Richardson, D. C. & Wang, B.-C. (2005). *J. Struct. Funct. Gen.* 6, 1-11.
- Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. & Bourne, P. E. (2000). *Nucl. Acids Res.* 28, 235-242.
- Brunger, A.T., Kuriyan, J. & Karplus, M. (1987). *Science* 235, 458-460.
- Chen, V.B., Arendall, W. B. III, Headd, J. J., Keedy, D. A., Immormino, R. M., Kapral, G. J., Murray, L. W., Richardson, J. S., and Richardson, D. C. (2009) *Acta Cryst. D*, in press.
- Davis, I. W., Leaver-Fay, A., Chen, V. B., Block, J. N., Kapral, G. J., Wang, X., Murray, L. W., Arendall, W. B., 3rd, Snoeyink, J., Richardson, J. S. & Richardson, D. C. (2007). *Nucleic Acids Res* 35, W375-383.
- DeLano, W.L. (2002) The PyMOL Molecular Graphics System. DeLano Scientific LLC, Palo Alto, CA, USA. <http://www.pymol.org>
- Emsley, P. & Cowtan, K. (2004). *Acta Cryst. D* 60, 2126-32.
- Flynn, G.E., Black, K.D., Islas, L.D., Sankaran, B. & Zagotta, W.N. (2007). *Structure* 15, 671-682.
- Gelbin, A., Schneider, B., Clowney, L., Hsieh, S.-H., Olson, W.K., & Berman, H.M.. (1996) *J. Am. Chem. Soc.* 118: 519-528.
- Grosse-Kunstleve, R. W., Sauter, N. K., Moriarty, N. W. & Adams, P. D. (2002). *J. Appl. Cryst.* 35, 126-136.
- Grosse-Kunstleve, R.W. & Adams, P.D. (2003). *Acta Cryst. D* 59, 1966-1973.
- Grosse-Kunstleve, R. W. & Brunger, A. T. (1999). *Acta Cryst. D* 55, 1568-1577.

- Headd, J. J., Immormino, R. M., Keedy, D. A., Emsley, P., Richardson, D. C. & Richardson, J. S. (2009). *J Struct Funct Genomics* 10, 83-93.
- Kostecky, B., Saurin, A.T., Purkiss, A., Parker, P.J. & McDonald, N.Q. (2009). *EMBO reports* 10, 983-989.
- Lebedev, A. A., Vagin, A. A. & Murshudov, G. N. (2006). *Acta Cryst. D*62, 83-95.
- Lovell, S. C., Word, J. M., Richardson, J. S. & Richardson, D. C. (2000). *Proteins* 40, 389-408.
- Lovell, S. C., Davis, I. W., Arendall, W. B. III, de Bakker, P. I. W., Word, J. M., Prisant, M. G., Richardson, J. S. & Richardson, D. C. (2003). *Proteins* 50, 437-450.
- Lutz, M. & Ascher, D. (1999). *Learning Python*. California, USA: O'Reilly & Associates.
- Padilla, J. E. & Yeates, T. O. (2003). *Acta Cryst. D*59, 1124-1130.
- Parkinson, G., Vojtechovsky, J., Clowney, L., Brunger, A.T., & Berman, H.M. (1996) *Acta Cryst. D*52: 57-64.
- Popov, A. N. & Bourenkov, G. P. (2003). *Acta Cryst. D*59, 1145-1153.
- McCoy, A.J., Storoni, L.C. and Read, R.J. (2004). *Acta Cryst. D* 60, 1220-1228.
- McCoy, A. J., Grosse-Kunstleve, R. W., Storoni, L. C. & Read, R. J. (2005). *Acta Cryst. D* 61, 458-464.
- McCoy, A. J. (2007). *Acta Cryst. D* 63, 32-41.
- McCoy, A.J., Grosse-Kunstleve, R.W., Adams, P.D., Winn, M.D., Storoni, L.C. and Read, R.J. (2007). *J. Appl. Cryst.* 40, 658-674 (2007).
- Miller, R., Gallo, S. M., Khalak, H. G. & Weeks, C. M. (1994). *J. Appl. Cryst.* 27, 613-621.
- Moriarty, N. W., Grosse-Kunstleve, R.W., Adams, P.D. (2009). *Acta Cryst. D* 65, 1074-1080.
- Morris, R. J., Zwart, P. H., Cohen, S., Fernandez, F. J., Kakaris, M., Kirillova, O., Vornrhein, C., Perrakis, A. & Lamzin, V. S. (2004). *J. Synchrotron Rad.* **11**, 56-59.
- Navaza, J. & Vernoslava, E. (1995). *Acta Cryst. A*51, 445-449.
- Read, R.J. (1999). *Acta Cryst. D*55, 1759-1764.
- Read, R.J. (2001). *Acta Cryst. D* 57, 1373-1382.
- Richardson, J.S., Schneider, B., Murray, L. W., Kapral, G. J., Immormino, R. M., Headd, J. J., Richardson, D. C., Ham, D., Herskovits, E., Williams, L. D., Keating, K. S., Pyle, A. M., Micallef, D., Westbrook, J. & Berman, H. M. (2008). *RNA* 14, 465-481.
- Schneider, T. R. & Sheldrick, G. M. (2002). *Acta Cryst. D*58, 1772-1779.
- Stewart, J. J. P. (2004). *J. Mol. Model.* **10**, 155-164.
- Storoni, L.C., McCoy, A.J. and Read, R.J. (2004). *Acta Cryst. D* 60, 432-438.
- Terwilliger, T. C. and Berendzen, J. (1999). *Acta Cryst. D* 55, 849-861.
- Terwilliger, T. C. (2000). *Acta Cryst. D* 55, 1863-1871.
- Terwilliger, T. C. (2001). *Acta Cryst. D* 57, 1755-1762.
- Terwilliger, T. C. (2002a). *Acta Cryst. D* 58, 2213-2215.
- Terwilliger, T. C. (2002b). *Acta Cryst. D* 58, 2082-2086.
- Terwilliger, T. C. (2003a). *Acta Cryst. D* 59, 38-44.
- Terwilliger, T. C. (2003b). *Acta Cryst. D* 59, 45-49.
- Terwilliger, T. C. (2003c). *Acta Cryst. D* 59, 1688-1701.



- Terwilliger, T. C. (2004). *Acta Cryst. D* 60, 2144-2149
- Terwilliger, T.C., Klei, H., Adams, P.D., Moriarty, N.W., Cohn, J.D. (2006). *Acta Cryst. D* 62, 915-922.
- Terwilliger, T.C., Adams, P.D., Moriarty, N.W., Cohn, J.D. (2007a). *Acta Cryst. D* 63, 101-107.
- Terwilliger, T.C., Grosse-Kunstleve, R.W., Afonine, P.V., Moriarty, N.W., Zwart, P.H., Hung, L.W., Read, R.J., Adams, P.D. (2007b). *Acta Cryst D* 64, 61-69.
- Terwilliger, T.C., Grosse-Kunstleve, R.W., Afonine, P.V., Adams, P.D., Moriarty, N.W., Zwart, P., Read, R.J., Turk, D., Hung, L.-W. (2007c). *Acta Cryst. D* 63, 597-610.
- Terwilliger, T.C., Grosse-Kunstleve, R.W., Afonine, P.V., Moriarty, N.W., Adams, P.D., Read, R.J., Zwart, P., Hung, L.-W. (2008). *Acta Cryst D* 64, 515-524.
- Terwilliger, T.C., Adams, P.D., Read, R.J., McCoy, A. J., Moriarty, N.W., Grosse-Kunstleve, R.W., Afonine, P.V., Zwart, P., Hung, L.-W. (2009). *Acta Cryst D* 65, 582-601.
- Urzhumtseva, L., Afonine, P.V., Adams P.D. & Urzhumtsev, A. (2009). *Acta Cryst. D* 65, 297-300.
- Wang, B.-C. (1985). *Methods Enzymol.* 115, 90-112.
- Wang, X., Kapral, G.J., Murray, L.W., Richardson, D.C., Richardson, J.S., & Snoeyink, J. (2008) *J. Math. Biol.* 56 (special RNA issue), 253-278.
- Word, J. M., Lovell, S. C., LaBean, T. H., Taylor, H. C., Zalis, M. E., Presley, B. K., Richardson, J. S. & Richardson, D. C. (1999a). *J. Mol. Biol.* 285, 1711-1733.
- Word, J. M., Lovell, S. C., Richardson, J. S. & Richardson, D. C. (1999b). *J. Mol. Biol.* 285, 1735-1747.
- Zwart, P.H. (2005). *Acta Cryst D* 61, 1437-1448.
- Zwart, P.H., Grosse-Kunstleve, R.W. and Adams, P.D. (2005). Xtriage and Fest: automatic assessment of X-ray data and substructure structure factor estimation. CCP4 newsletter Winter, Contribution 7.
- Zwart, P.H., Grosse-Kunstleve, R.W., Lebedev, A.A., Murshudov, G.N. & Adams, P.D. (2008). *Acta Cryst. D* 64, 99-107.